EpiBuffer Examples

```r
##########################
# DEPENDENCIES
##########################
library(EpiBuffer) # SingleBuffer and SpatialBuffers functions
library(dplyr)     # various helper functions
library(tidyr)     # various helper functions
library(fields)    # various helper functions
library(ggplot2)   # plotting
library(HDInterval) # hdi function
```

Function to simulate outcomes, radii, exposure effect

```r
sim_data_function <- function(exposure_indicator,
                              unique_dists,
                              unique_ps_dists,
                              max_radius,
                              trials,
                              x,
                              w,
                              v,
                              beta_true,
                              gamma_true,
                              eta_true,
                              rho_phi_true,
                              sim_setting,
                              seed){

  ################
  # Global Settings
  ################
  n_ind <- nrow(x)
  n_unique <- ncol(v)

  spatial_cov <- exp(-rho_phi_true*unique_dists)

  phi_true <- mnormt::rmnorm(n = 1,
                             mean = 0.00,
                             varcov = spatial_cov)

  # extract w and phi for each individual based on location
  v_w <- w[v, , drop = FALSE]         # (n_ind x p_w)
  v_phi <- phi_true[v , drop = FALSE] # (n_ind x 1)


  if(sim_setting == 0){ # no effect (theta = 0)
    radius_trans_true <- rep(0.00, times = n_ind)
    theta_true <- rep(0.00, times = n_ind)

  }

  # True theta and radius (depends on sim setting)
  if(sim_setting == 1){ # single radius, single theta

    radius_trans_true <- rep(gamma_true[1], times = n_ind)
    theta_true <- rep(eta_true[1], times = n_ind)

  }

  if(sim_setting == 2){ # varying radii, single theta

    radius_trans_true <- (v_w%*%gamma_true) + v_phi
    theta_true <- rep(eta_true[1], times = n_ind) # theta = eta0 (pd = 0 setting)

  }

  if(sim_setting == 3){ # varying radii, varying theta

    radius_trans_true <- (v_w%*%gamma_true) + v_phi
    radius_true <- max_radius*pnorm(radius_trans_true)

    # theta is coefficient on the radius (not on radius_trans!)
    theta_true <- cbind(1, (radius_true))%*%eta_true # pd = 1 setting

  }

  radius_true <- as.vector(max_radius*pnorm(radius_trans_true))

  # True exposure
  exposure_true <- rep(NA, times = n_ind)
  if(exposure_indicator == 0){ # counts
    v_exposure_dists <- unique_ps_dists[v, , drop = FALSE]
    exposure_true <- rowSums(v_exposure_dists < radius_true)
  }
  if(exposure_indicator == 2){ # presence/absence
    # exposed if at least one facility is within distance radius_true
    for(j in 1:n_ind){
      exposure_true[j] <- max(
        as.numeric(unique_ps_dists[c(1:n_unique)[v[j,] == 1],] <= radius_true[j])
      )
    }
  }

  ###########
  # Outcome
  ###########
  # Simulate outcome from Binomial Distribution
  # logit(p) = log(p / 1-p) = x*\beta + \theta*exposure
  logit_p <- x%*%beta_true + theta_true*exposure_true
```

```r
  probs <- (1.00/(1.00 + exp(-logit_p)))
  y <- rbinom(n = n_ind,
              size = trials,
              p = probs)

  ################
  # Return sim dat
  ################
  return(list(y = y,
              radius_true = radius_true,
              theta_true = theta_true,
              exposure_true = exposure_true,
              probs_true = probs))

}
```

Create inputs for sim data function

Specify simulation parameters

```r
# Set seed for reproducability
seed <- 1234

# Set all global simulation parameters
n_ind <- 500              # total number of outcome units
n_ind_unique <- 108       # total number of unique locations
m <- 100                   # total number of exposure sources
max_radius <- 20          # maximum radius to consider
exposure_indicator <- 0   # 0: counts
```

```r
# True parameter values
beta_true <- c(-0.95, 0.33)
rho_phi_true <- 3.07
gamma_true <- c(0.11, 0.07)
eta_true_single <- 0.90
```

Simulate location and covariate information

```r
set.seed(seed)

# Unique locations of outcome units
unique_locs <- data.frame(
  location_id = 1:n_ind_unique,
  x = runif(n_ind_unique, min = 0, max = max_radius*5),
  y = runif(n_ind_unique, min = 0, max = max_radius*5)
)

# Randomly assign outcome units to unique locations
ind_locs <- data.frame(
  individual_id = 1:n_ind,
  location_id = c(
    sample(1:n_ind_unique, size = n_ind_unique, replace = FALSE),  # guarantee one per location
    sample(1:n_ind_unique, size = n_ind - n_ind_unique, replace = TRUE) # sample remaining
  )
)

# Sample locations of exposure sources (``ps")
ps_locs <- data.frame(
  ps_id = 1:m,
  x = runif(m, min = 0, max = max_radius*5),
  y = runif(m, min = 0, max = max_radius*5)
)

#.............Create Location Matrixes................#

# v: index vector indicating unique location row in exposure_dists for each outcome unit
# length: n_ind
# NOTE: v maps each individual to a row in exposure_dists and w;
# rows must be ordered consistently with location indexing:
# row i corresponds to location_id[i], where location_id[v[j]] is the location for outcome unit j.
v <- match(ind_locs$location_id, unique_locs$location_id)

# Grid: knot locations for predictive process approximation
# here - same as unique locations (can be different or a subsample, as desired)
grid_coords <- unique_locs[match(unique_locs$location, unique_locs$location_id),
                           c("location_id", "x", "y")]
grid_coords$location_id <- paste0("grid_", grid_coords$location_id)

all_locs <- rbind(unique_locs, grid_coords)

# unique_ps_dists: pairwise distance matrix for all unique locations and exposure sources
# only retain necessary exposure sources based on max_radius
unique_ps_dists <- fields::rdist(unique_locs[,c("x", "y")], ps_locs[,c("x", "y")])
unique_ps_dists <- unique_ps_dists[, !apply(unique_ps_dists > max_radius, 2, all)]

# unique_dists: pair-wise distance matrix for all unique locations
# dim: (n_ind_unique x n_ind_unique)
unique_dists <- fields::rdist(unique_locs[,c("x", "y")], unique_locs[,c("x", "y")])
unique_dists <- unique_dists / max(unique_dists)

# full_dists: pair-wise distance matrix for all unique locations and grid points
# dim: (n_ind_unique + n_grid) x (n_ind_unique + n_grid)
full_dists <- fields::rdist(all_locs[,c("x", "y")], all_locs[,c("x", "y")])
full_dists <- full_dists / max(full_dists) # scale to between 0 and 1

#.............Create Covariate Data................#
# trials: individuals per location
# modeling Bernoulli data - each individual is a trial
trials <- rep(1, n_ind)

# x: covariate matrix
# dim: nInd x p_x
x <- data.frame(
  x0 = rep(1, n_ind),
  x1 = rbinom(n_ind, size = 1, prob = 0.5) # a factor covariate
) %>% as.matrix()
```

```r
# w: spatial covariate matrix (for SpatialBuffer)
# dim: n_unique x p_w
w <- data.frame(
  intercept = rep(1, n_ind_unique),
  w1 = rnorm(n_ind_unique, mean = 0, sd = 1)
) %>% as.matrix()
```

Create simulated data sets from different settings

Recall -

- sim_setting = 0: no effect;
- sim_setting = 1: single radius, single effect;
- sim_setting = 2: varying radius, varying effect;
- sim_setting = 3: varying radius, varying effect

```r
# Simulated data where true dgp is single radius, single effect
sim_dat1 <- sim_data_function(exposure_indicator = exposure_indicator,
                              unique_dists = unique_dists,
                              unique_ps_dists = unique_ps_dists,
                              max_radius = max_radius,
                              trials = trials,
                              x = x,
                              w = w,
                              v = v,
                              beta_true = beta_true,
                              gamma_true = gamma_true,
                              eta_true = eta_true_single,
                              rho_phi_true = rho_phi_true,
                              sim_setting = 1,
                              seed = seed)

# Simulated data where true dgp is varying radius, single effect
sim_dat2 <- sim_data_function(exposure_indicator = exposure_indicator,
                              unique_dists = unique_dists,
                              unique_ps_dists = unique_ps_dists,
                              max_radius = max_radius,
                              trials = trials,
                              x = x,
                              w = w,
                              v = v,
                              beta_true = beta_true,
                              gamma_true = gamma_true,
                              eta_true = eta_true_single,
                              rho_phi_true = rho_phi_true,
                              sim_setting = 2,
                              seed = seed)
```

Fit Models

Set Global MCMC parameters

```
mcmc_samples <- 50000
burnin <- 30000
thin <- 2
keep_set <- seq(burnin, mcmc_samples, by = thin)
```

True DGP: Single Radius, Single Effect (sim_setting = 1)

SingleBuffer

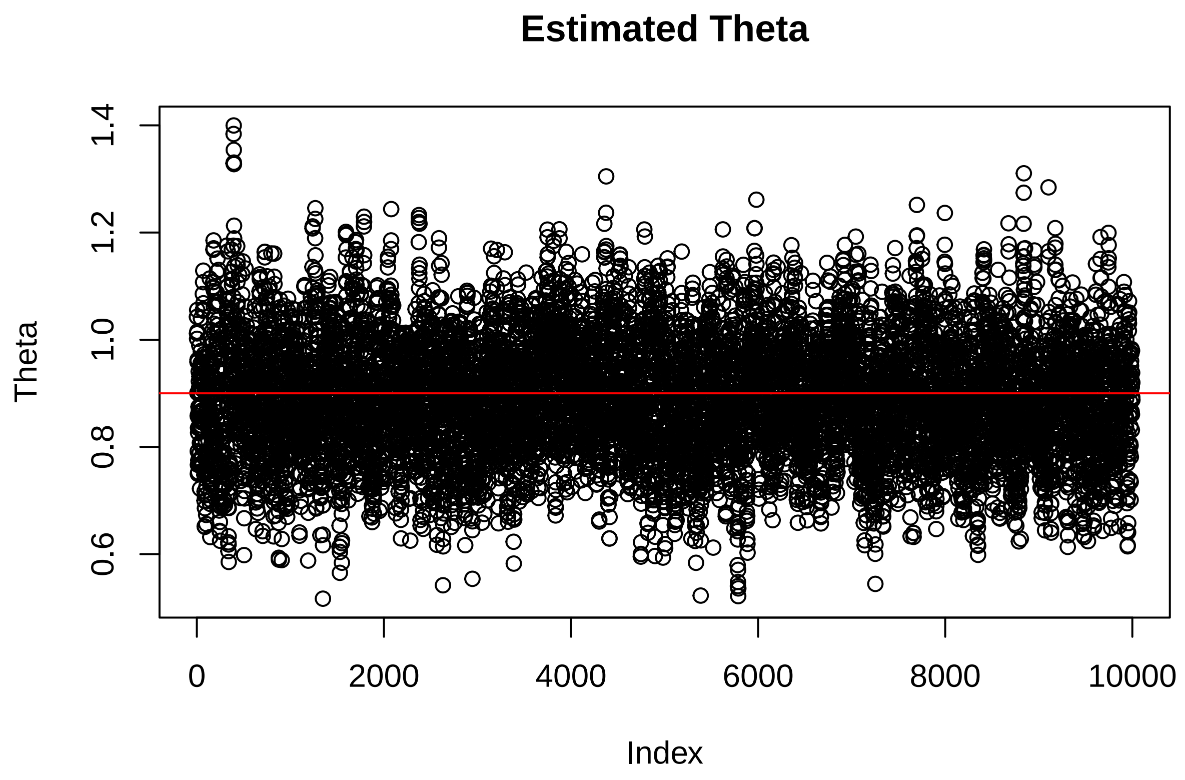
```
print("Fitting SINGLEBUFFER")
```

```
## [1] "Fitting SINGLEBUFFER"
```

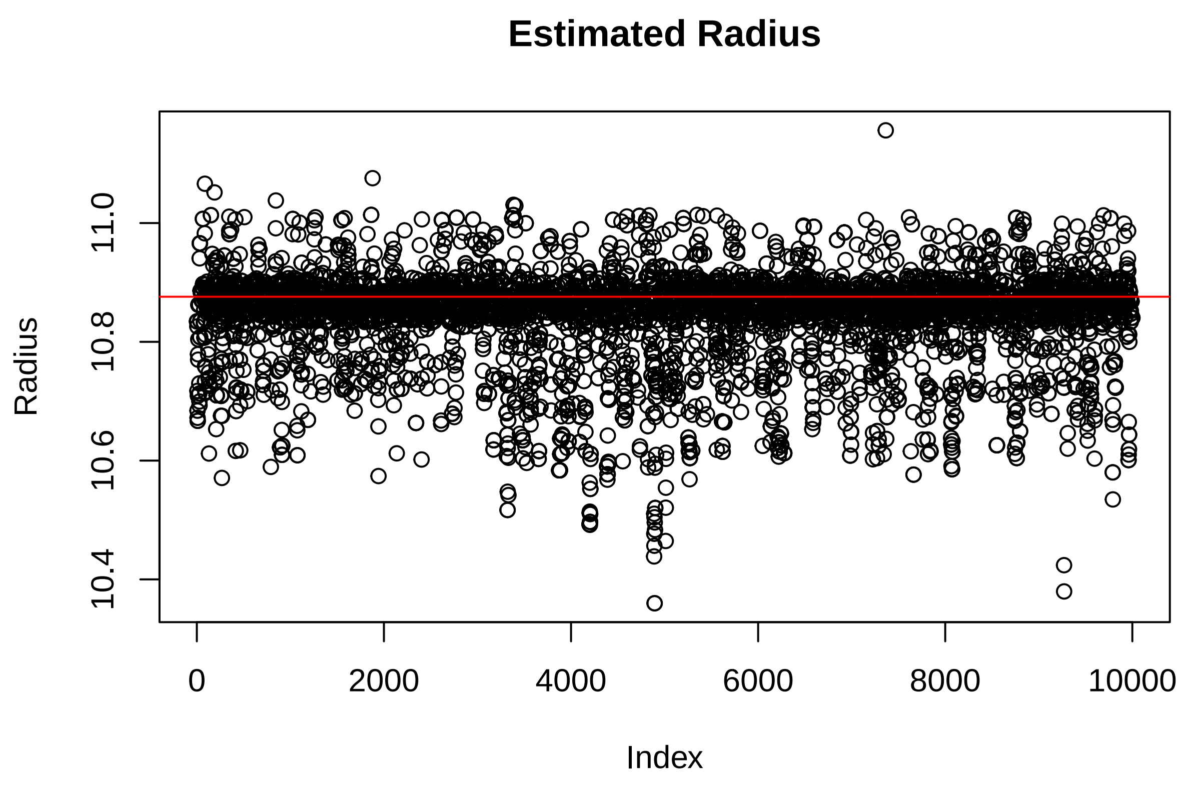
```
metrop_var_radius <- 0.001

set.seed(seed)
res_single <- EpiBuffer::SingleBuffer(mcmc_samples = mcmc_samples,
                                      y = sim_dat1$y,
                                      x = x,
                                      v = v,
                                      radius_range = c(0.00, max_radius),
                                      exposure_definition_indicator = exposure_indicator,
                                      exposure_dists = unique_ps_dists,
                                      metrop_var_radius = metrop_var_radius,
                                      likelihood_indicator = 0,
                                      trials = trials,
                                      waic_info_indicator = 1)
```

```
## Progress: 10%
## radius Acceptance: 23%
## **********************
## Progress: 20%
## radius Acceptance: 23%
## **********************
## Progress: 30%
## radius Acceptance: 23%
## **********************
## Progress: 40%
## radius Acceptance: 23%
## **********************
## Progress: 50%
## radius Acceptance: 23%
## **********************
## Progress: 60%
## radius Acceptance: 24%
## **********************
## Progress: 70%
## radius Acceptance: 23%
## **********************
## Progress: 80%
## radius Acceptance: 23%
## **********************
## Progress: 90%
## radius Acceptance: 23%
## **********************
## Progress: 100%
## radius Acceptance: 23%
## **********************
```

```r
# Plot for theta
plot(res_single$theta[1, keep_set] / res_single$exposure_scale,
     main = "Estimated Theta",
     ylab = "Theta",
     xlab = "Index")
abline(h = unique(sim_dat1$theta_true), col = "red", lty = 1)
```



```r
# Plot for radius
plot(res_single$radius[keep_set],
     main = "Estimated Radius",
     ylab = "Radius",
     xlab = "Index")
abline(h = unique(sim_dat1$radius_true), col = "red", lty = 1)
```



True DGP: Varying Radius, Varying Effect (sim_setting = 3)

SpatialBuffer, pd = 1

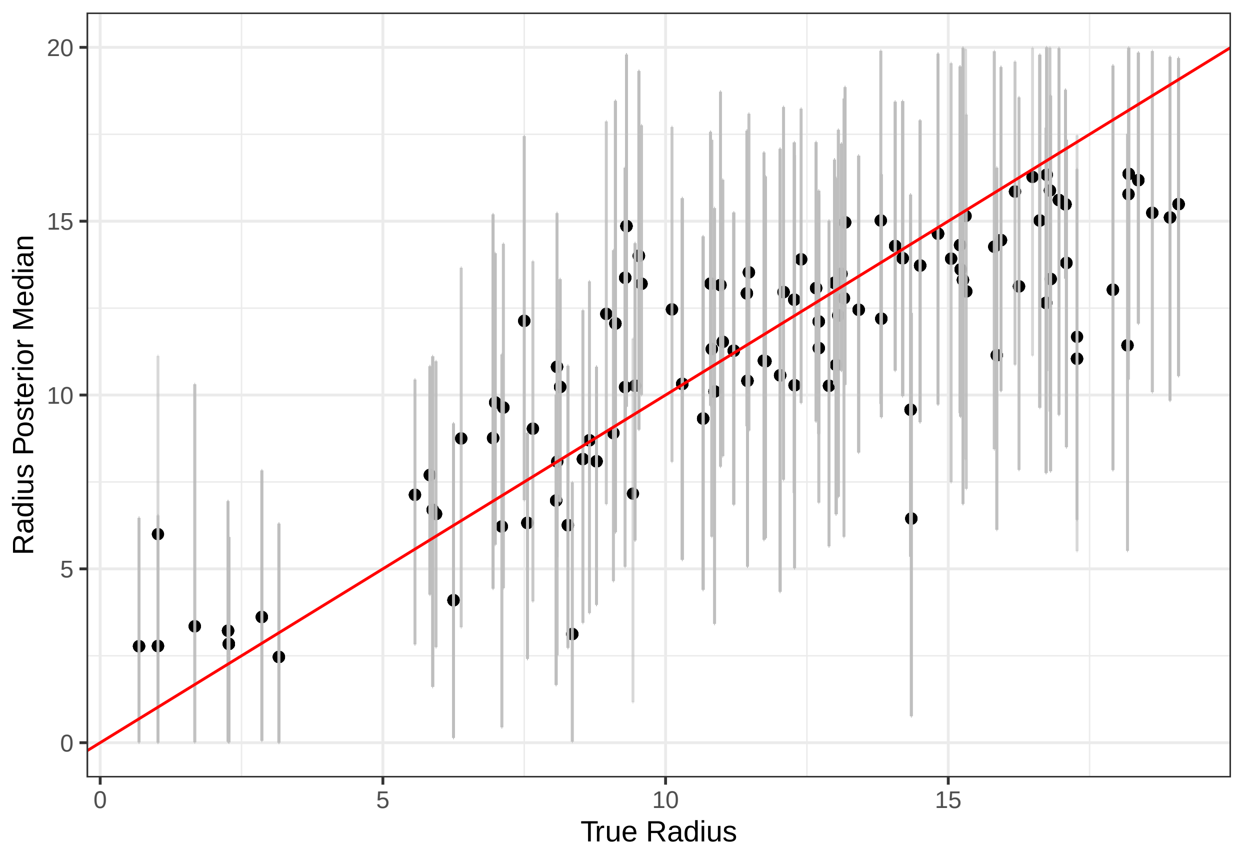
```
print("Fitting SPATIALBUFFER, pd = 0")
```

```
## [1] "Fitting SPATIALBUFFER, pd = 0"
```

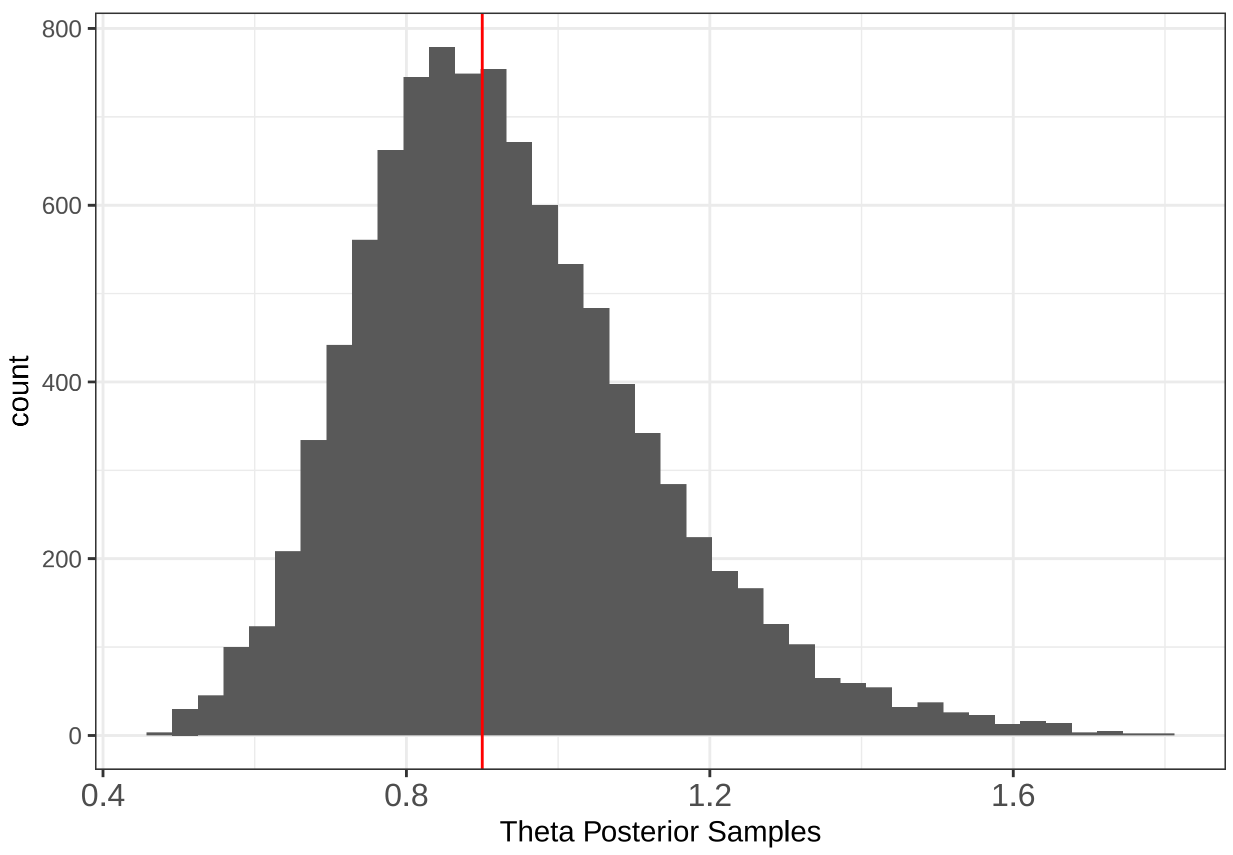
```
metrop_var_gamma <- 0.01
metrop_var_phi_star <- 0.2
metrop_var_rho_phi <- 1.0

set.seed(seed)
res_spatial1 <- EpiBuffer::SpatialBuffers(mcmc_samples = mcmc_samples,
                                          y = sim_dat2$y,
                                          x = x,
                                          w = w,
                                          v = v,
                                          radius_range = c(0.00, max_radius),
                                          exposure_definition_indicator = exposure_indicator,
                                          exposure_dists = unique_ps_dists,
                                          p_d = 0,
                                          full_dists = full_dists,
                                          metrop_var_gamma = rep(metrop_var_gamma,
                                                                 times = ncol(w)),
                                          metrop_var_phi_star = rep(metrop_var_phi_star,
                                                    times = length(v)),
                                          metrop_var_rho_phi = metrop_var_rho_phi,
                                          likelihood_indicator = 0,
                                          trials = trials,
                                          waic_info_indicator = 1)
```

```
## Progress: 10%
## gamma Acceptance (min): 14%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 30%
## phi_star Acceptance (max): 66%
## rho_phi Acceptance: 18%
## ******************************
## Progress: 20%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 30%
## phi_star Acceptance (max): 65%
## rho_phi Acceptance: 18%
## ******************************
## Progress: 30%
## gamma Acceptance (min): 14%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 30%
## phi_star Acceptance (max): 66%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 40%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 30%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 18%
## ******************************
## Progress: 50%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 30%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 60%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 31%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 70%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 31%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 80%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 31%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 90%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 31%
## phi_star Acceptance (max): 68%
## rho_phi Acceptance: 17%
## ******************************
## Progress: 100%
## gamma Acceptance (min): 13%
## gamma Acceptance (max): 16%
## phi_star Acceptance (min): 31%
## phi_star Acceptance (max): 67%
## rho_phi Acceptance: 17%
## ******************************
```

```r
# Extract output and plot results
radius_df <- data.frame(t(res_spatial1$radius)) # Rows = MCMC iterations; columns = outcome units
radii_thinned <- radius_df[keep_set,]
radius_post_summary <- radii_thinned %>%
  dplyr::summarise(across(everything(), list( # apply to each column (individual)
    posterior.median = ~median(.),
    posterior.mean = ~mean(.),
    hdi.lower95 = ~hdi(., credMass = 0.95)[["lower"]],
    hdi.upper95 = ~hdi(., credMass = 0.95)[["upper"]],
    ci.lower95 = ~quantile(., 0.025),
    ci.upper95 = ~quantile(., 0.975)
  ))) %>%
  pivot_longer(cols = everything(), names_to = c("cluster", ".value"), names_sep = "_")
radius_post_summary$radius_true <- sim_dat2$radius_true

theta_df <- data.frame(t(res_spatial1$theta))
theta_df <- theta_df/res_spatial1$exposure_scale # unscale thetas
theta_thinned <- theta_df[keep_set, ]

ggplot(radius_post_summary,
       aes(x = radius_true, y = posterior.median)) +
  geom_point() +
  geom_errorbar(aes(ymin = hdi.lower95, ymax = hdi.upper95), # Add credible intervals
                width = 0.01,
                color = "grey", alpha = 0.6) +
  xlab("True Radius") +
  ylab("Radius Posterior Median") +
  theme_bw()+
  geom_abline(slope = 1, intercept = 0, color = "red")
```



```r
ggplot(theta_thinned, aes(x = `X1`)) +
  geom_histogram(bins = 40) +
  geom_vline(xintercept = sim_dat2$theta_true, color = "red") +
  xlab("Theta Posterior Samples") +
  theme_bw() +
  theme(axis.text.x=element_text(size=12))
```


count
800
600
400
200
0
0.4
0.8
1.2
1.6
Theta Posterior Samples